

Date⇒ 01-02-2021

Module⇒ Backend

Lecture By⇒ Akash Handa

Subject ⇒ Crud

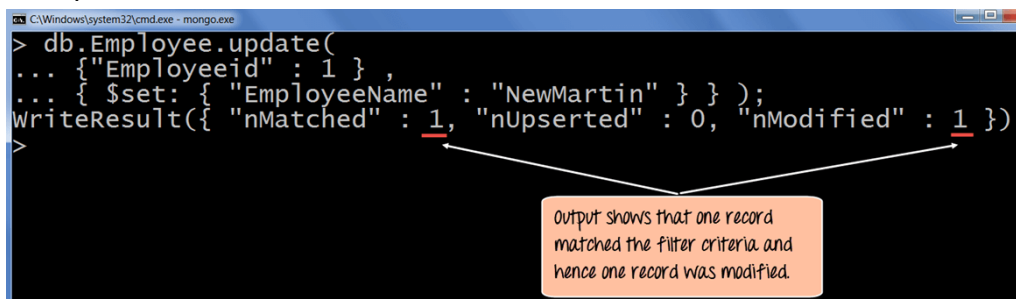
IN PREVIOUS LECTURE (QUICK RECAP) Date-29/1/2021	In Today's Lecture (Overview)
How to Connect Node.js File with mongo data base Import MongoClient Create our main function	Update command in mongodb Delete command in mongodb Upsert in Mongodb Crud Operation in Mongodb Create Operations Read Operations Update Operations Delete Operations

Update command in mongodb

Modifies an existing document or documents in a collection. The method can modify specific fields of an existing document or documents or replace an existing document entirely, depending on the [update parameter](#).

By default, the `db.collection.update()` method updates a **single** document. Include the option **multi: true** to update all documents that match the query criteria.

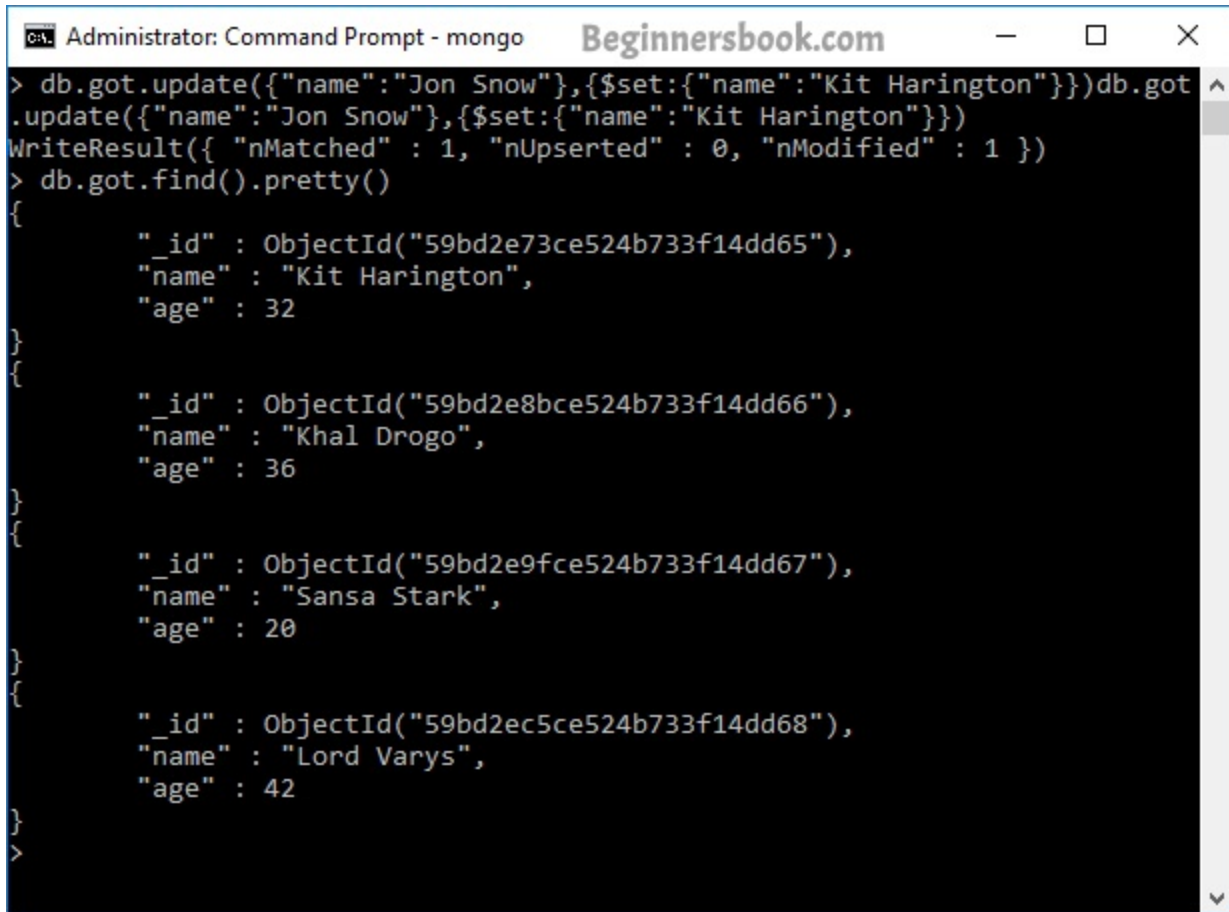
example=1



```
C:\Windows\system32\cmd.exe - mongo.exe
> db.Employee.update(
... {"Employeeid" : 1 },
... { $set: { "EmployeeName" : "NewMartin" } } );
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Output shows that one record matched the filter criteria and hence one record was modified.

example=2



```
> db.got.update({'name':'Jon Snow'},{$set:{'name':'Kit Harington'}})db.got
.update({'name':'Jon Snow'},{$set:{'name':'Kit Harington'}})
WriteResult({ 'nMatched' : 1, 'nUpserted' : 0, 'nModified' : 1 })
> db.got.find().pretty()
{
  "_id" : ObjectId("59bd2e73ce524b733f14dd65"),
  "name" : "Kit Harington",
  "age" : 32
}
{
  "_id" : ObjectId("59bd2e8bce524b733f14dd66"),
  "name" : "Khal Drogo",
  "age" : 36
}
{
  "_id" : ObjectId("59bd2e9fce524b733f14dd67"),
  "name" : "Sansa Stark",
  "age" : 20
}
{
  "_id" : ObjectId("59bd2ec5ce524b733f14dd68"),
  "name" : "Lord Varys",
  "age" : 42
}
>
```

Delete command in mongodb

Removes documents from a collection.

The `db.collection.remove()` method can have one of two syntaxes. The `remove()` method can take a query document and an optional `justOne` boolean:

example=1

```
Administrator: Command Prompt - mongo
> use beginnersbookdb
switched to db beginnersbookdb
> db.students.find().pretty()
{
  "_id" : ObjectId("59bcecc7668dcce02aaa6fed"),
  "StudentId" : 1001,
  "StudentName" : "Steve",
  "age" : 30
}
{
  "_id" : ObjectId("59bcecc7668dcce02aaa6fef"),
  "StudentId" : 3333,
  "StudentName" : "Rick",
  "age" : 35
}
> db.students.remove({"StudentId": 3333})
WriteResult({ "nRemoved" : 1 })
> db.students.find().pretty()
{
  "_id" : ObjectId("59bcecc7668dcce02aaa6fed"),
  "StudentId" : 1001,
  "StudentName" : "Steve",
  "age" : 30
}
>
```

example=2

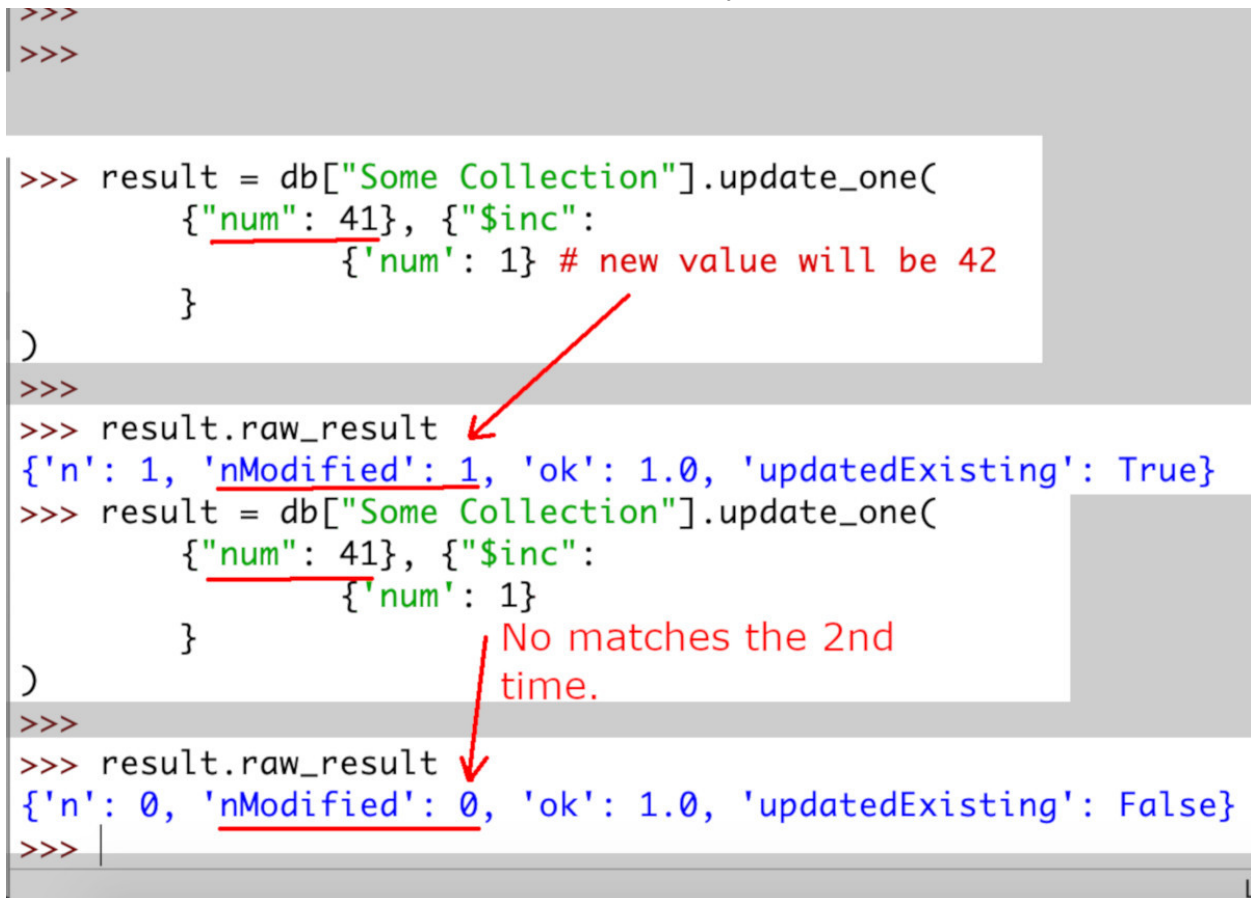
```
Python 3.7.3 Shell
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 16:52:21)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> from pymongo import MongoClient
>>> mongo_client = MongoClient('mongodb://localhost:27017')
>>> db = mongo_client["Some-Database"]
>>>
>>> db['Some Collection'].find_one({"field 4": "value 4"})
{'_id': ObjectId('5cf8c6838dbf4d4c309c3f39'), 'field 4': 'value 4'}
>>> result = db['Some Collection'].delete_one( {"doc does not": "exist"} )
>>> print (result, "-- deleted_count:", result.deleted_count)
<pymongo.results.DeleteResult object at 0x107fd4d48> -- deleted_count: 0
>>>
>>> from bson import ObjectId
>>> result = db['Some Collection'].find_one( { "_id" : ObjectId("5cf8c6498dbf4d4c309c3f36") } )
>>> print (result, "-- type:", type(result))
{'_id': ObjectId('5cf8c6498dbf4d4c309c3f36'), 'field 1': 'value 1'} -- type: <class 'dict'>
>>> result = db['Some Collection'].delete_one( { "_id" : "5cf8c6838dbf4d4c309c3f39" } )
>>> print ("delete count:", result.deleted_count)
delete count: 0
>>> result = db['Some Collection'].delete_one( { "_id" : ObjectId("5cf8c6838dbf4d4c309c3f39") } )
>>> print ("delete count:", result.deleted_count)
delete count: 1
>>>
```

Upsert in MongoDB

MongoDB upsert option is used with update method which creates a new document if the query does not retrieve any documents satisfying the criteria. The default value for this option is false. The upsert option does an insert based on the field and value pairs specified in the update parameter or field and value pairs specified in both query and update parameter.

```
>>>
>>>

>>> result = db["Some Collection"].update_one(
    {"num": 41}, {"$inc":
        {'num': 1} # new value will be 42
    }
)
>>>
>>> result.raw_result
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
>>> result = db["Some Collection"].update_one(
    {"num": 41}, {"$inc":
        {'num': 1}
    }
)
>>>
>>> result.raw_result
{'n': 0, 'nModified': 0, 'ok': 1.0, 'updatedExisting': False}
>>>
```



No matches the 2nd time.

Crud Operation in MongoDB

- Create Operations
- Read Operations
- Update Operations
- Delete Operations

Create Operations

Create or insert operations add new [documents](#) to a [collection](#). If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- `db.collection.insertOne()` *New in version 3.2*
- `db.collection.insertMany()` *New in version 3.2*

In MongoDB, insert operations target a single [collection](#). All write operations in MongoDB are [atomic](#) on the level of a single [document](#).

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,            ← field: value
  status: "pending"   ← field: value
}                    } document
)
```

Read Operations

Read operations retrieve [documents](#) from a [collection](#); i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- `db.collection.find()`

You can specify [query filters or criteria](#) that identify the documents to return.

```
db.users.find(
  { age: { $gt: 18 } },
  { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

Update Operations

Update operations modify existing [documents](#) in a [collection](#). MongoDB provides the following methods to update documents of a collection:

- `db.collection.updateOne()` *New in version 3.2*
- `db.collection.updateMany()` *New in version 3.2*
- `db.collection.replaceOne()` *New in version 3.2*

In MongoDB, update operations target a single collection. All write operations in MongoDB are [atomic](#) on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These [filters](#) use the same syntax as read operations.

```
db.users.updateMany(
  { age: { $lt: 18 } },
  { $set: { status: "reject" } }
)
```

← collection
← update filter
← update action

Delete Operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- `db.collection.deleteOne()` *New in version 3.2*
- `db.collection.deleteMany()` *New in version 3.2*

In MongoDB, delete operations target a single [collection](#). All write operations in MongoDB are [atomic](#) on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These [filters](#) use the same syntax as read operations.

```
db.users.deleteMany(  ← collection  
  { status: "reject" } ← delete filter  
)
```